Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра информационно-аналитических систем

Кокаия Мария Платоновна

Анализ базы данных траекторий автомобилей

Дипломная работа

Научный руководитель:
д. ф.-м. н., профессор Новиков Б. А.

Рецензент:
ст. преп. Шпильман А. А.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Chair of Analytical Information Systems

Kokaiia Mariia

Analysis of vehicle trajectory database.

Graduation Thesis

Scientific supervisor:
professor Novikov Boris

Reviewer:
lecturer Shpilman Aleksei

Saint-Petersburg
2016

Оглавление

АННОТАЦИЯ

В данной дипломной работе были определены экономические предпочтения автомобилистов Москвы на примере заправок. Необходимо было проанализировать базу данных с gps координатами траекторий водителей, а также некоторыми показателями вождения.

В результате работы по имеющимся данным для каждой заправки была выделена траектория движения автомобиля, которая с большой вероятностью гарантирует, что этой заправкой воспользуются. Кроме того, был разработан алгоритм, позволяющий для произвольной траектории движения определять подходящие заправки. И построена модель, определяющая наилучшие места для установки новых заправок. Резльтаты были визуализированы с помощью 2GIS API.

ВВЕДЕНИЕ

В настоящее время определить целевую аудиторию и выявить причины экономических предпочтений покупателей не так сложно. Количество людей, осознанно или случайно пополняющих Интернет данными, постоянно растет. Существует огромный объем, анализируя которые можно многое узнать о пользователях – где они работают, что предпочитают, какие товары их интересуют, узнать в целом о человеческом поведении.

В данной дипломной работе была поставлена цель - определить экономические предпочтения автомобилистов Москвы на примере заправок. Была выдана база данных с gps координатами траекторий водителей, а также некоторыми показателями вождения.

Для достижения цели были поставлены следующие задачи:

1. Для каждой заправки определить траекторию движения автомобиля, которая с большой вероятностью гарантирует, что этой заправкой воспользуются.

2. Разработать алгоритм, позволяющий для произвольной траектории движения определять подходящие заправки.

3. Построить модель определяющую наилучшие места для установки новых заправок.

Не существует единого алгоритма, который бы смог по любым данным классифицировать объекты, предсказывать поведение и решать

другие задачи области анализа данных. Часто то, что нужно анализировать представлено в виде на первый взгляд не пригодном для анализа. В таких случаях необходимо провести дополнительную обработку данных, которая может оказаться не менее сложной задачей. Особенностью работы стало то, что необходимо было анализировать траектории для определения экономических предпочтений. Для решения этой задачи была построена модель на графах.

Разработка велась на языке Python, база данных MS SQL.

1. ОБЗОР ЛИТЕРАТУРЫ

Для предсказаний связанных с дорожным движением часто используют оценки поведения водителей, например рискованность движения, резкие повороты, количество обгонов, нарушений и т.д. Кроме того, учитывают ситуацию на дороге: загруженность, время суток (The Impact of Driver Inattention on Near-Crash/Crash Risk: An Analysis Using the 100-Car Naturalistic Driving Study Data, [1]). Но это только часть информации. Для выявления экономических предпочтений полезно анализировать и сами траектории.

Для анализа траекторий и анализа движения было разработано большое количество различных методов. Обобщая их, можно выделить два основных шага:

1. упрощение каждой траектории;

2. выявление общих закономерностей путем сравнения и группировки траекторий.

При анализе траектории как геометрической линии, на практике для ее упрощения часто используется алгоритм Дугласа-Пекера [2] который позволяет удалить часть точек, сохраняя при этом прежнюю геометрическую форму. Кроме того, траектории дополнительно можно разбить на подгруппы (Lee, J.G., Han, J., and Whang, K.Y., Trajectory clustering: a partitionandgroup framework, [3]) и последующий анализ будет в первую очередь сосредоточен на подтраекториях. Для большого количества данных этот подход работает медленно.

Альтернативный подход фокусируется на анализе фигур, которые ограничивают траектрии (A graph-based approach to vehicle trajectory analysis, Diansheng Guo, [4]). Таким образом геометрическое упрощение заменяется топологическим. Для измерения сходства между траекториями после упрощения извлекается вектор признаков [5], в который входят такие атрибуты траекторий, как скорость движения, продолжительность, кривизна и другие. Извлеченные атрибуты могут быть обработаны с расчетом метрики подобия (Tiakas, E., Searching for similar trajectories in spatial networks [6]) и в зависимости от задачи использованы далее, например, для метода опорных векторов (Dodge, S., Weibel, R., and Forootan, E., Revealing the physics of movement: comparing the similarity of movement characteristics of different types of moving objects, [5]).

Подход, применяющийся в дипломной работе отличается от имеющихся. На первом шаге траектории упрощаются с помощью кластерного анализа (A branch and bound algorithm for computing k-nearest neighbors, K Fukunaga, [7]). Этот подход часто применяется для анализа географических данных, но редко для траекторий. Центры кластеров становятся новыми точками, определяющими траекторию.

Для их анализа поверх сетки траекторий построен граф. Вектор признаков в таком случае состоит из весов ребер в графе, которые интерпретируются в вероятность того, что автомобиль воспользуется конкретным путем. Такой подход не хранит ни геометрическую, ни топологическую форму движения, но сохраняет вероятностную модель и выделяет основные точки движения. Это достаточная информация для определения экономических предпочтений. Получившиеся данные являются обучающими для алгоритмов классификации. В дан-

ном случае использовался показывающий хорошие результаты алгоритм RandomForest [8] и часто используемая на практике Logistic Regression [9].

2. ОПИСАНИЕ ИСХОДНЫХ ДАННЫХ

2.1. ФОРМАТ ДАННЫХ

Для анализа данных выдан дамп базы в .bak файле. С помощью Microsoft SQL Server Managment Studio база была восстановлена. Для дальнейшей работы таблицы выкачаны в текстовые файлы, чтобы их можно было обработать на операционной системе Ubuntu.

В целом данные представляют информацию о поездках по Москве 1459 автомобилей. Траектории движения автомобилей разбиты на некоторые отрезки (trip), которые в свою очередь состоят из наборов точек, которые зафиксировал GPS спутник.

Данные состоят из следующих таблиц:

1. points - данные gps, точки движения автомобилей. В дальнейшем были представлены как траектории. Столбцы: идентификатор машины; широта точки с GPS; долгота точки с GPS; скорость в данной точке; высота в данной точке; направление движения объекта; время.

2. scores - таблица с оценками поездки. Столбцы: суммарное количество баллов, полученных за резкие ускорения; суммарное количество баллов, полученных за резкие торможения; суммарное количество баллов, полученных за резкие повороты; всего штрафных баллов набрано; рисковость поездки; оценка поездки.

3. trip - информация о поездке. Столбцы: время начала; время конца; максимальна скорость; длительность поездки; протяженность поездки.

4. trip_positions - координаты начала и конца поездки.

2.2. ВЫДЕЛЕНИЕ ТРАЕКТОРИЙ

Для решения поставленных в работе задач были выделены траектории движения автомобилистов, которые проходят через заправки, более того, этими заправками пользовались.

Для начала необходимо было получить информацию в целом о координатах заправок Москвы. Для подобных запросов существует несколько API, например openstreetmap или 2GIS. В работе данные получены у сервиса Google Places API. Google Places API Web Service – это служба, из которой с помощью HTTP-запросов можно получать информацию о следующих местах: географических объектах, организациях или достопримечательностях. Google Places API принимает запросы и возвращает данные в формате JSON или XML. При этом в запросах обязательно должен использоваться протокол https://, а также должен быть указан ключ API. [17]

Для поиска координат заправок в Москве у Google Places API есть подходящие запросы поиска мест, возвращающие список близлежащих мест с учетом условий запроса, позволяет запрашивать информацию об организациях, достопримечательностях, географических объектах и

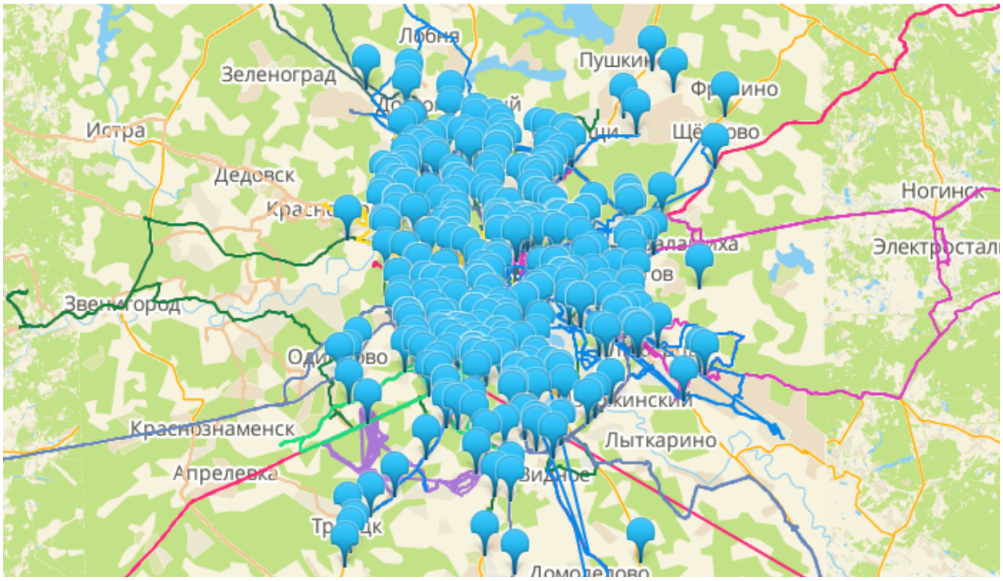

Рис. 1: Заправки Москвы

многих других местах на карте. Есть возможность поиска места в заданном радиусе или по прямому текстовому запросу. Возвращается список мест и общая информация о каждом из них. Дополнительные данные можно получить с помощью запроса данных о месте.

В результате были получены координаты 1776 заправок. В данных, предоставляемых Google Places API, заправка - это координаты одной точки. Машинами, заезжающими на заправку считались все, находящиеся в радиусе 10 метров от заправки. Такой поиск определил 921 машину из 1459.

Для более точной выборки была использована информация о размерах некоторых заправок Москвы и посчитан их средний радиус. Такая выборка должна удовлетворять следующим условиям:

1. радиус заправки: 32 метра;

2. водитель провел на заправке от 3 до 15 минут.

Таким условиям удовлетворяло 1135 машин и 5682 траектории.

2.3. ИСПОЛЬЗОВАНИЕ MAPREDUCE

Объем базы с координатами в текстовом формате составил 700 GB. Для обработки данных такого объема лучше всего воспользоваться готовыми инструментами. В работе использован алгоритм MapReduce. Он позволяет уменьшить время выполнения задачи, а также уменьшить количество используемой памяти, что является актуальной проблемой, для обработки данных на среднем по мощности ПК.

Для поиска оптимальной траектории к заправке необходимо было определить идентификаторы машин, заезжавших на эту заправку. В рамках алгоритма MapReduce [10] задача решалась следующим образом:

1. На Map-шаге происходит предварительная обработка входных данных. Главный узел получает входные данные задачи, разделяет их на части и передает рабочим узлам. Шаг подобен операции map в функциональных языках программирования – пользовательская функция применяется к каждой входной записи. Результат - это множество пар ключ-значение. Ключ очень важен, так как данные с одним ключом в будущем попадут в один экземпляр функции Reduce.

   Для каждой машины ставились в соответствие заправки, которыми автомобиль пользовался. Для этого в несколько потоков считывались соответствующие файлы с координатами движения каждой машины. В Табл.1 показан пример получившегося списка.

Таблица 1: Соответствие машин и заправок.

car_id	gas_station_id
100	22
100	25
...	...

2. На Shuffle-шаге вывод функции map «разбирается по корзинам» – каждая корзина соответствует одному ключу вывода стадии Map. В дальнейшем эти корзины послужат входом для Reduce.

3. На Reduce-шаге происходит свёртка предварительно обработанных данных. Главный узел получает ответы от рабочих узлов и на их основе формирует результат.

Для текущей задачи для каждой уникальной заправки в соответствие ставились идентификаторы автомобилей.

Кроме того:

1. Все запуски функции Map работают независимо и могут работать параллельно.

2. Все запуски функции Reduce работают независимо и могут работать параллельно.

3. МОДЕЛЬ НА ГРАФАХ

Для определения вероятной траектории и экономических предпочтений траектории были представлены в виде некоторой модели. Геометрическая форма движения не была сохранена. Но учитывалась частота, с которой курсируют машины, а также направления движения. Моделью, удовлетворяющей этим условиям, стала модель на графах.

3.1. ПОСТРОЕНИЕ ГРАФА ВЕРОЯТНОСТЕЙ

В работе основной моделью используемой для выявления экономических предпочтений автомобилистов является граф. Это структура, которая сохраняет координаты "важных" (вероятных) в движении точек, их связь и вероятность перемещения из одной точки в другую. Граф строится для каждой заправки, основываясь на точках поездок, использующих эту заправку.

Алгоритм построения графа следующий:

1. Для каждой заправки выбираются поездки, проходящие через эту заправку. Затем из данных GPS выбираются точки, относящиеся к этим поездкам.

2. С помощью алгоритма поиска k ближайших соседей, где k равно 100, точки собираются в кластера. Это позволяет облегчить модель. Точки, которые находятся на небольшом друг от друга расстоянии, будут представлены как одна, в данном случае такой точкой будет центр кластера.

3. Точки, получившиеся после кластеризации становятся вершинами графа.

4. Ребро между вершинами проводится в том случае, если любые точки из двух кластеров последовательно входят в одну поездку.

5. Граф должен быть взвешен. Вес ребра означает вероятность с которой автомобиль переместится из одной вершины в другую. Рассчитывается количество перемещений в определенную вершину для каждого входящего в нее ребра, таким образом вес ребра: количество перемещений по этому ребру к вершине к общему числу всех перемещений в эту вершину.

3.2. ПОСТРОЕНИЕ ВЕРОЯТНОЙ ТРАЕКТОРИИ

После того, как граф построен, необходимо определить траекторию движения автомобиля, которая с большой вероятностью гарантирует, что конкретной заправкой воспользуются.

Для решения этой задачи необходимо сделать обход графа от вершины равной координатам заправки, по всем следующим, выбирая максимальный вес ребра на каждом шаге.

3.3. ОПРЕДЕЛЕНИЕ ПРЕДПОЧТЕНИЙ

Построенный граф позволяет определять подходящие заправки для произвольной траектории движения. Для этого по известным координатам новой траектории были выделены все заправки в радиусе 100

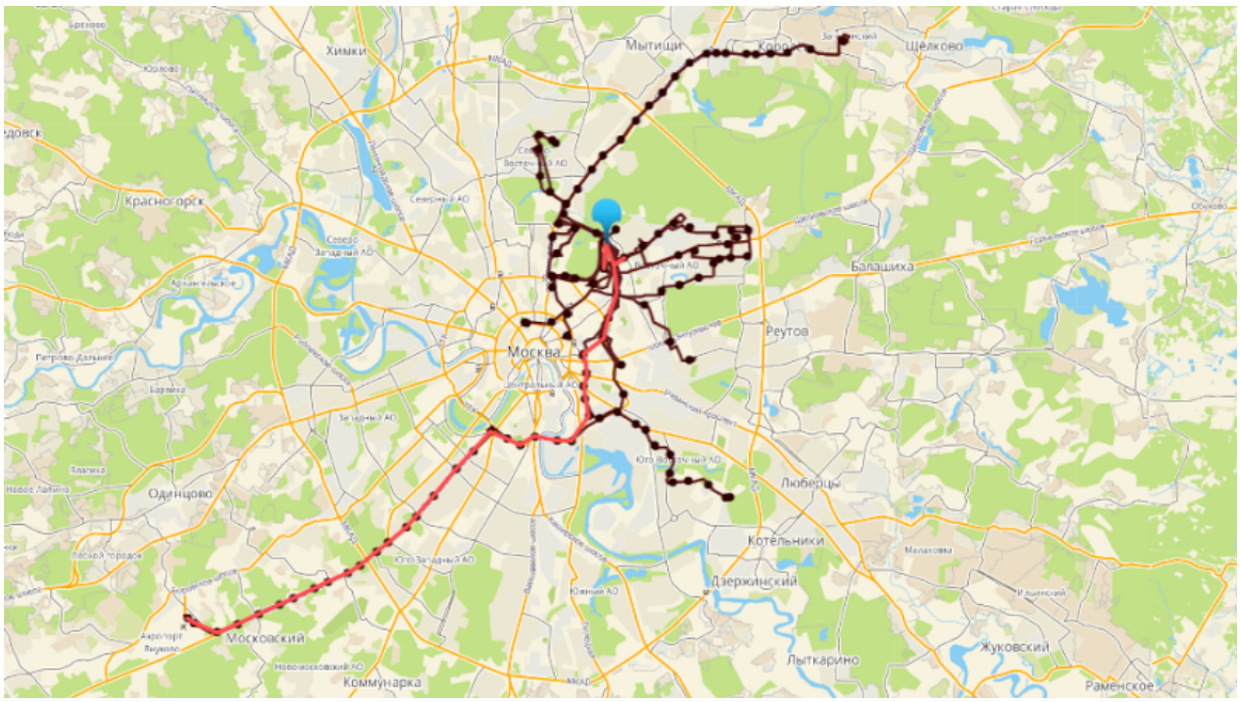

Рис. 2: Вероятная траектория

метров и для каждой посчитана вероятность того, что автомобиль воспользуется именно этой заправкой.

Для того, чтобы рассчитать вероятность пользования конкретной заправкой было придумано следующее решение:

1. были найдены близкие или совпадающие точки с вершинами графа;

2. для этих вершин находились соответствующие ребра;

3. вероятность для конкретной заправки выражалась как произведение весов всех найденных ребер.

Результат работы алгоритма представлен на рисунке. Траектория движения автомобиля совпала с 10 точками графа, оставшиеся 5 точек

графу не принадлежали. Вероятность того, что автомобиль заедет на заправку, оказалась равна 0.69%.

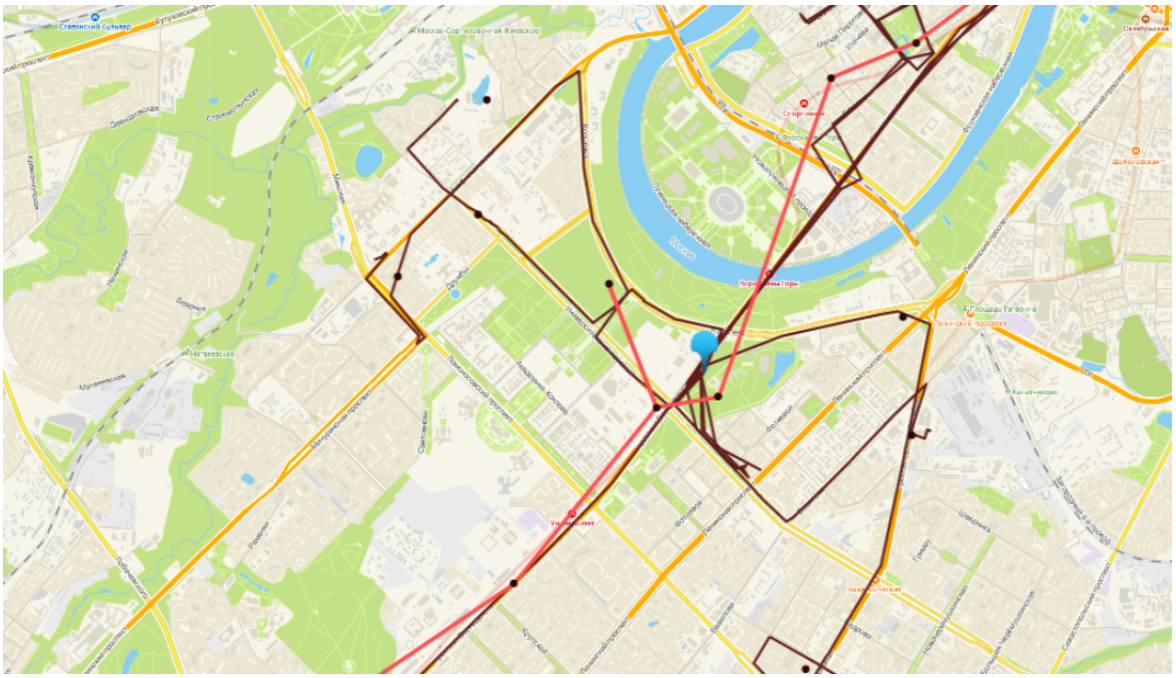

Рис. 3: Построение графа для произвольной траектории

4. ОПРЕДЕЛЕНИЕ МЕСТ ДЛЯ ЗАПРАВОК

Для определения наилучших мест для установки новых заправок необходимо построить модель, которая с помощью методов машинного обучения будет предсказывать устанавливать на этом месте заправку или нет. Подходящими для анализа данными являются данные из таблицы scores, которая содержит информацию об оценках поездки и gps координаты движения автомобилистов.

4.1. ПОСТРОЕНИЕ ОБОБЩАЮЩЕГО ГРАФА

Для того, чтобы представить информацию о траекториях в виде вектора признаков, были использованы уже построенные графы заправок. Для этого графы были представлены в виде матриц.

Как показано в таблице, где “N” - обозначение вершины.

	“1”	“2”	“3”	“4”	“5”
“1”	0	0.7	0.3	0	0
“2”	0	0	0	1	0
“3”	0	0	0	0	1

Таблица 2: Графы в виде матриц

Затем, была построена модель объединяющая эти матрицы, по следующему алгоритму:

1. Для каждого графа посчитать количество вершин на каждом слое, где слой определяется как количество вершин, которые нужно пройти от некоторой вершины до вершины-заправки.

2. Объединяющая матрица строится из максимальных значений на каждом слое.

Таким образом вектор признаков состоит из весов ребра объединенного графа. Так, для каждой заправки матрица представляется в виде объединенной, где, если у графа ребро отсутствует, то его вес равен 0.

Для имеющихся данных возникла следующая проблема: все ребра для объединенного графа считаются признаками, таким образом для одной заправки из некоторой вершины выходит N ребер, а для остальных заправок из этой вершины выходит N2 ребра, где N2 меньше N. Тогда получается, что значения для (N - N2) признаков (ребер) кроме одного случая нулевые. Таким образом, около 30% имели описанную выше структуру. Решение подобной проблемы в дополнительной обработке обобщающей матрицы, удалении таких ребер и пересчете значений для остальных ребер.

4.2. КЛАССИФИКАЦИЯ

Задача о поиске оптимальных решений является типичной задачей классификации. Где таргет функция принимает два значения: есть заправка или нет. В качестве алгоритмов классификации в работе выбраны RandomForest, KNeighborsClassifier, LogisticRegression и SVM из пакета Sklearn для языка Python. Алгоритмы выбраны не случайно, в настоящее время это самые популярные алгоритмы, используемые в рабочих системах.

4.2.1. СЛУЧАЙНЫЙ ЛЕС

sklearn.ensemble.RandomForestClassifier

В машинном обучении не редки случаи, когда построенный классификатор слишком слаб, и не удовлетворяет по качеству. В подобных случаях возникает вопрос, возможно ли агрегировать такие классификаторы в один, который будет классифицировать точно.

Постановка задачи выглядит следующим образом.

Имеется обучающая выборка:

$$X^l = (x_i, y_i)_{i=1}^l, x_i \in X, y_i \in \{-1, +1\} \tag{1}$$

И некоторое количество базовых классификаторов:

$$b_1(x), ..., b_T(x), b_t : X \mapsto \{-1, +1\} \tag{2}$$

Простым голосованием называется такой классификатор, который относит объект X к тому классу к которому его отнесли большинство базовых классификаторов.

$$a(x) = \sum_{t=1}^{T} b_t(x) \tag{3}$$

Существует несколько способов повышения различности базовых алгоритмов. Одим из самых известных способов является бэггинг. В этом методе алгоритмы $b_t(x)$ обучаются независимо по случайным подвыборкам с повторениями. [14]

Случайный лес - это специальный случай бэггинга, когда в качестве базовых классификаторов используются решающие деревья. Построение деревьев весьма специфично: в ходе создания очередного узла дерева признак, на основе которого производится разбиение, выбирается не из всех, а из m случайно выбранных. Выбор наилучшего из этих m признаков может осуществляться различными способами. Чаще всего используется критерий Джини. Он показывает сколько есть пар объект лежащих в одном и том же классе, которые вместе идут либо в левую дочернюю вершину либо в правую.

$$I(b, X^l) = \#\{(x_i, x_j) : y_i = y_j b(x_i) = b(x_j)\} \tag{4}$$

То есть критерий меряет на сколько часто объекты одних классов объединяются. [13]

Таким образом, работа алгоритма заключается в следующем: строится некоторое количество подвыборок с повторениями и для каждой строится дерево. Оптимальное число деревьев подбирается таким образом, чтобы на тестовой выборке минимизировать ошибку классификатора.

На рисунке ниже представлен пример дерева, классифицирующего данные заправок.

Классификация объектов проводится путём голосования: каждое дерево относит классифицируемый объект к одному из классов, и побеждает класс, за который проголосовало наибольшее число деревьев.

model_rfc = RandomForestClassifier(n_estimators = 70) В параметре передаем количество деревьев.

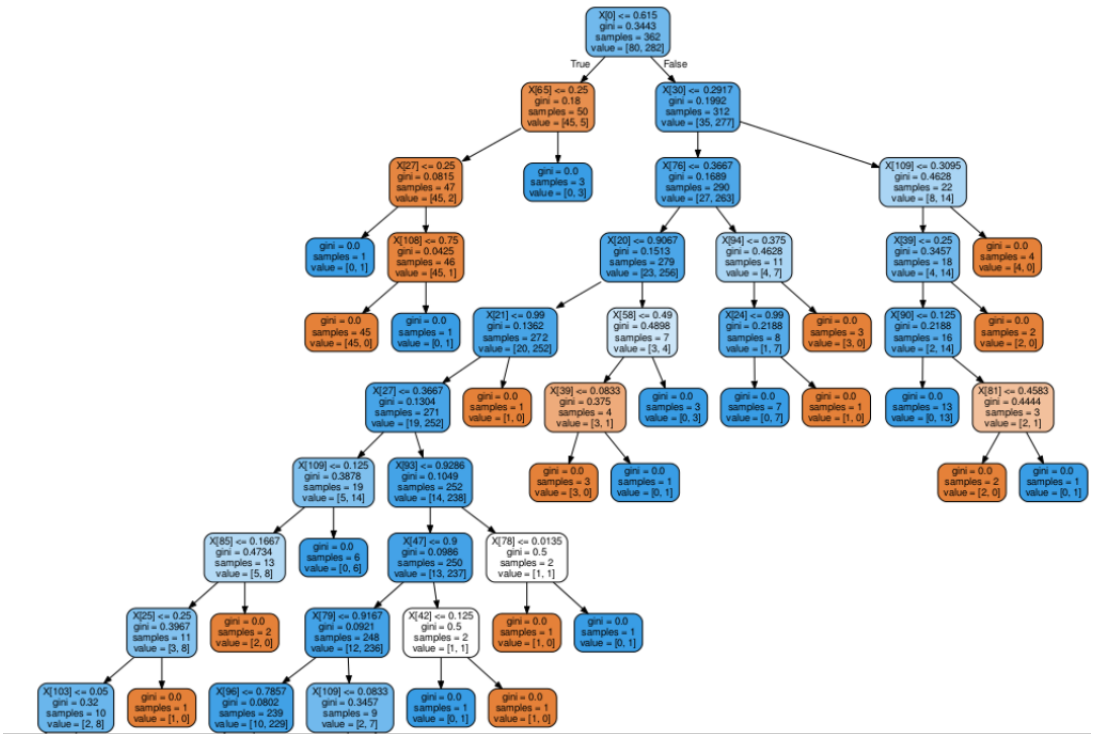

Рис. 4: Дерево, классифицирующее данные заправок.

4.2.2. МЕТОД К БЛИЖАЙШИХ СОСЕДЕЙ

sklearn.neighbors.KNeighborsClassifier

Метод kNN (kNearest Neighbors) часто используется как дополнение для более сложного алгоритма классификации. Нередко результаты его работы используются в качестве признака для объекта.

Формальная постановка задачи.

Дано множество объектов $X \in R^n$.

Множество допустимых ответов Y.

Задана обучающая выборка $\{(x_i, y_i)\}_{i=1}^{l}$.

Задано множество объектов $X^m = \{x_i\}_{i=1}^{m}$.

Требуется найти множество ответов $\{y_i\}_{i=1}^{m}$.

На множестве объектов задается некоторая мера расстояния, например Евклидова $p(x, x^{|})$:

$$p(x, x^{|}) = \sum_{i=1}^{n} (x_i - x_i^{|})^2 \tag{5}$$

Тот объект обучающей выборки, который является i-м соседом объекта x обозначим за $x_{i;x}$.

Аналогичное для ответа на i-м соседе: $y_{i;x}$.

Тогда алгоритм ближайших соседей есть

$$a(x) = \arg\max_{y} \sum_{i=1}^{m} [x_{i;x} = y] w(i, x), \tag{6}$$

Весовая функция, которая оценивает степень важности i-го соседа для классификации объекта w(i,x).

Так, при $w(i, x) = 1$ и при $i < k$ алгоритм соответствует медоду k ближайших соседей. [18]

model_knc = KNeighborsClassifier(n_neighbors = 18)

4.2.3. ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

sklearn.linear_model.LogisticRegression

Математическая модель алгоритма следующая: вводится зависимая переменная y, принимающая одно из двух значений — как правило, это числа 0 (событие не произошло) и 1 (событие произошло), в контексте

задачи установлена заправка или нет, и множество независимых переменных $x_1, x_2, .., x_n$ на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной.

Делается предположение о том, что вероятность наступления события $y = 1$ равна:

$$P\{y = 1|x\} = f(z), \tag{7}$$

где $z = \theta^t x = \theta_1 x_1 + ... + \theta_n x_n$,

x — векторы-столбцы значений независимых переменных $x_1..x_n$

θ — векторы-столбцы параметров (коэффициентов регрессии) $\theta_1...\theta_n$

f(z) — логистическая функция (иногда также называемая сигмоидом или логит-функцией):

$$f(z) = \frac{1}{1 + e^{-z}} \tag{8}$$

Так как y принимает значения 0 и 1, то вероятность второго возможного значения равна:

$$P\{y = 0|x\} = 1f(z) = 1 - (\theta^T x)[19] \tag{9}$$

LogisticRegression(penalty='l1', tol=0.01)

4.2.4. МЕТОД ОПОРНЫХ ВЕКТОРОВ

sklearn.svm.SVC

Формально задача формулируется следующим образом:

Пусть каждый объект данных представлен как вектор (точка) в p-мерном пространстве. Полагаем, что точки имеют вид:

$$(x_1, c_1), (x_2, c_2)..., (x_n, c_n) \tag{10}$$

где c_i принимает значение 1 или -1, в зависимости от того, какому классу принадлежит точка x_i.

Каждое x_i — это p-мерный вещественный вектор, обычно нормализованный значениями [0,1] или [-1,1]. Необходимо построить разделяющую гиперплоскость следующего вида:

$$w * x - b = 0 \tag{11}$$

Вектор w — перпендикуляр к разделяющей гиперплоскости.

Параметр $\frac{b}{||w||}$ определяет смещение гиперплоскости относительно начала координат вдоль нормали w.

Если обучающие данные являются линейно разделимыми, то можно построить две гиперплоскости, которые разделяют данные так, что точек между ними не будет. Затем, необходимо максимизировать расстояние между ними.

Области, ограниченной 2 гиперплоскостями могут быть описаны уравнениями:

$$w * x - b = 1 \tag{12}$$

$$w * x - b = -1 \tag{13}$$

Расстояние между этими гиперплоскостями - $\frac{2}{||w||}$.

Для того чтобы дистанция была максимальной необходимо минимизировать $||w||$.

Таким образом, идея алгоритма состоит в поиске оптимально разделяющего классификатора. Каждый объект данных представляет собой вектор в некотором p—мерном пространстве. Соответственно классифицируя объект можно определить к какому из двух классов он относится. То есть задача формулируется так: можно ли разделить точки гиперплоскостью размерности $(p—1)$. Таких гиперплоскостей может быть много, но очевидно, что максимизируя зазор между классами, можно добиться более уверенной классификации. То есть необходимо найти такую гиперплоскость, чтобы расстояние от неё до ближайшей точки было максимальным. Таким образом, основная идея заключается в переводе исходных векторов в пространство более высокой размерности, и в дальнейшем поиске разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей. Алгоритм работает в предположении, что тем меньше будет средняя ошибка классификатора, чем больше расстояние между этими параллельными гиперплоскостями.

4.3. АНАЛИЗ РЕЗУЛЬТАТОВ

В качестве таргет функции выбрано бинарное поле, показывающее есть заправка или нет.

target = data.is_gas

Набор данных, на основании которого будет строиться модель, включает в себя оценки поездки и признаки (ребра) графа траекторий.

train = data.drop(['is_gas'], axis=1)

Результат проверен построением ROC кривых. Такой график обычно используется для визуализации результатов бинарной классификации. Данные делятся на положительные примеры и отрицательные, и предполагается, что должны быть верно классифицированы положительные примеры, и верно классифицированы отрицательные, но на практике возможны следующие исходы:

1. True Positives: положительные примеры были классифированны верно (истинно положительные случаи);

2. True Negatives: отрицательные примеры были классифицированны верно (истинно отрицательные случаи);

3. False Negatives: положительные примеры были классифицированны неверно (ложно отрицательные примеры);

4. False Positives: отрицательные примеры были классифицированны неверно (ложно положительные случаи).

ROC кривая определяет зависимость количества верно классифицированных положительных примеров от количества неверно классифицированных отрицательных примеров.

Для начала построим модели основываясь только на данных оценок вождения.

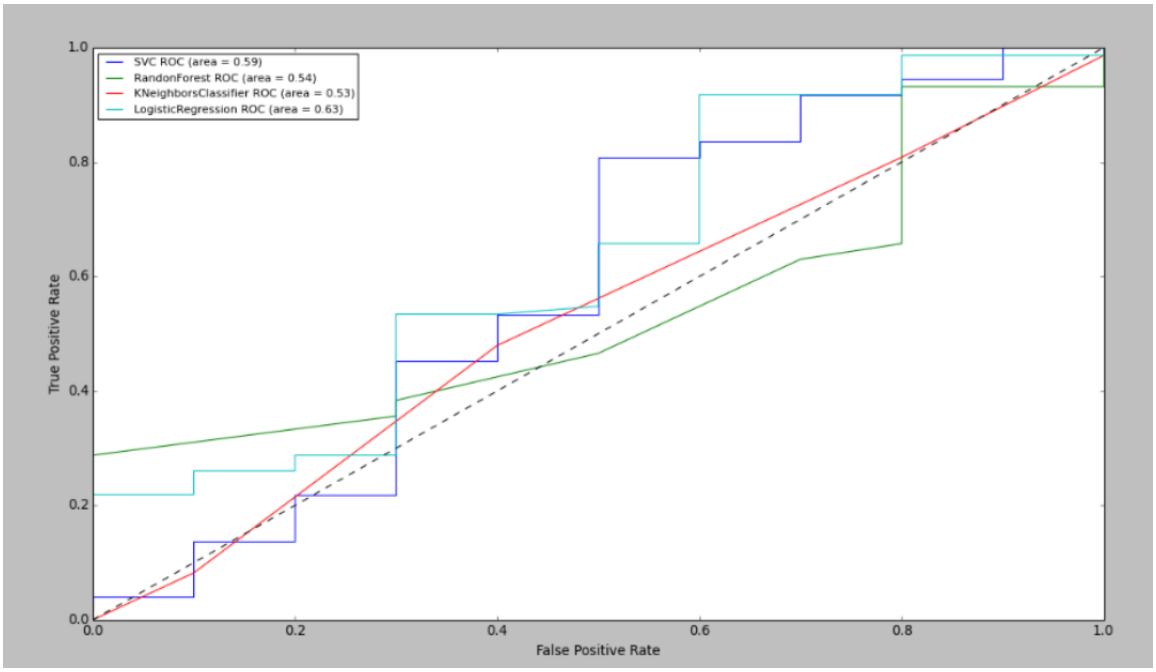

Рис. 5: ROC кривые для моделей по данным оценок вождения.

Для того, чтобы классификатор считался хорошим, график ROC кривой должен проходить там, где доля истинно положительных примеров составляет 100%, а доля ложно положительных равна нулю. Поэтому, чем меньше изгиб ROC кривой и чем ближе она к диагонали, тем худший результат показывает модель. Диагональная линия идентифицирует бесполезный классификатор, т.е. неспособный различить два класса.

Из графика видно, что все модели несколько раз пресекают диагональ, что говорит о том, что классификатор плохо обучен, и поэтому

бесполезен.

Построим модели, добавив данные графа.

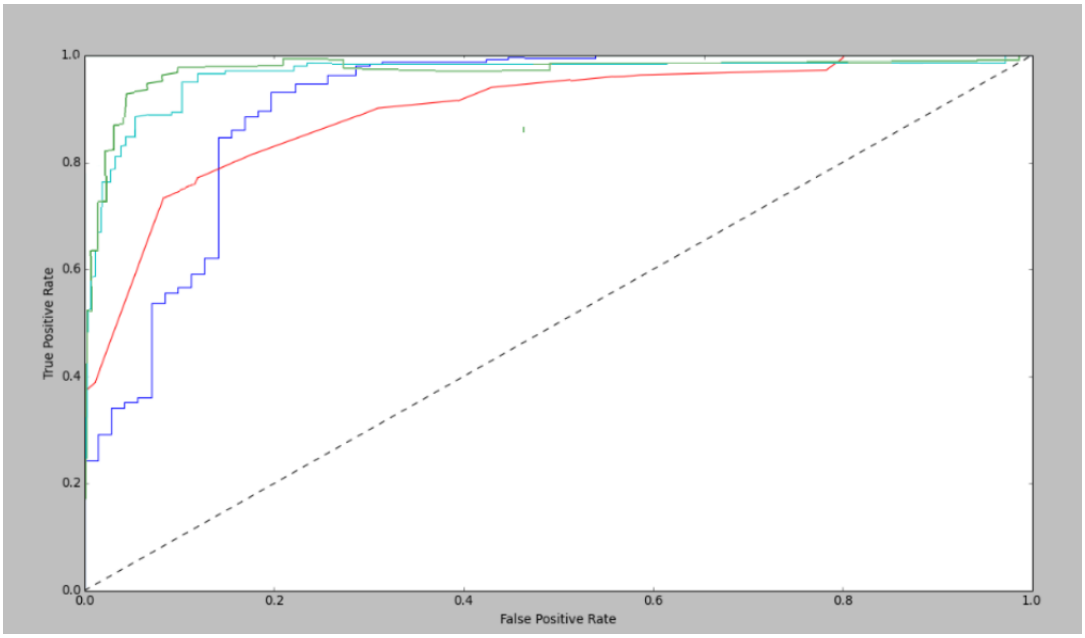

Рис. 6: ROC кривые для моделей по данным оценок вождения и данным графа.

Видно, что результаты улучшились. Кривые отдалились от диагонали.

Для того, чтобы понять место, где заправка будет располагаться удачно для выбранной точки строится граф (выданные данные о поездках покрывают всю Москву) и применяется получившуюся модель, в данном случае RandomForest, так как она показал лучший результат.

С помощью метода predict по полученной модели можно предсказать место, где заправки будут располагаться удачно, а где нет. На рисунке ниже представлены результаты для нескольких произвольных точек.

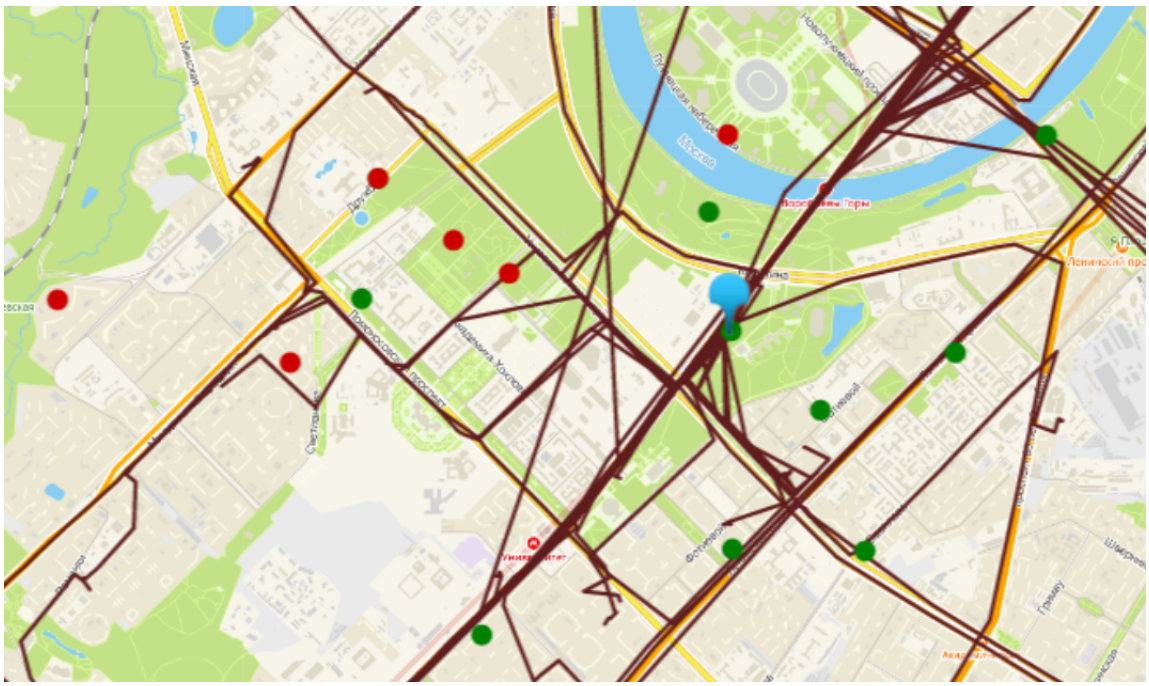

Рис. 7: Поиск наилучших мест для установки новых заправок, результаты полцчены классификатором RandomForest. Красным обозначены неудачные места. Зеленым удачные.

Результаты оказались не всегда справедливыми, так как не были учтены некоторые особенности. В частности, после того, как заправка будет установлена, часть машин изменят свои траектории по необходимости. Учитывая этот факт, можно сказать, что модель хорошо предсказывает удачные места для установки заправки, а на результаты неудач лучше не обращать внимания.

Еще одна ситуация, при которой модель может оказаться не точной, когда точка для установки новой заправки окажется рядом с уже имеющейся. В таком случае количество траекторий, проходящих через новую заправку будет больше обычного, но не из-за того, что это удачное место, а в следствии того, что там уже располагается заправка.

ЗАКЛЮЧЕНИЕ

Данная работа посвящена выявлению экономических предпочтений автомобилистов, в частности выявление предпочитаемых ими заправок. Для ее реализации были выданы не обычные для подобной задачи данные: не было ни информации о марке автомобиле, никакой информации об автомобилистах, только траектории движения и оценки вождения за некоторый промежуток времени. Получилась задача Data Mining об обнаружении в данных ранее неизвестных знаний.

В результате работы была придумана модель в виде графа, позволяющая представлять траектории движения в формате удобном для дальнейшего анализа. Это структура, которая сохраняет координаты “важных” (вероятных) в движении точек, их связь и вероятность перемещения из одной точки в другую.

Такая модель позволила для каждой заправки определить траекторию движения автомобиля, которая с большой вероятностью гарантирует, что этой заправкой воспользуются. Кроме того, основываясь на полученных графах можно определять траектории движения для произвольных траекторий движения автомобилистов в Москве.

Проанализировав имеющиеся данные, был построен классификатор, способный определять наилучшие места для установки новых заправок.

Список литературы

[1] National Highway Traffic Safety Administration (NHTSA). *The Impact of Driver Inattention on Near-Crash/Crash Risk: An Analysis Using the 100 Car Naturalistic Driving Study Data.* U.S. department of transportation, 2006.

[2] D. Douglas and T. Peucker. *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature.* 1973.

[3] Han J. Lee, J.-G. and K.-Y. Whang. *Trajectory clustering: a partition-andgroup framework.* 2007.

[4] Diansheng Guo. *A graph-based approach to vehicle trajectory analysis.* 2010.

[5] Weibel R. Dodge, S. and E. Forootan. *Revealing the physics of movement: comparing the similarity of movement characteristics of different types of moving objects.* 2009.

[6] et al. Tiakas, E. *Searching for similar trajectories in spatial networks.* 2009.

[7] K Fukunaga. *A branch and bound algorithm for computing k-nearest neighbors.* 1975.

[8] M. Wiener A. Liaw. *Classification and regression by RandomForest.* 2002.

[9] S Lemeshow DW Hosmer Jr. *Applied logistic regression.* 2004.

[10] Jeffrey Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters.* 2004.

[11] C. Cortes and V. Vapnik. *Support Vector Machine.* 1995.

[12] Goadrich M. Davis J. *The Relationship Between Precision-Recall and ROC Curves.* 2006.

[13] Воронцов К.В. Coursera. *Введение в машинное обучение. Алгоритм построения решающего дерева.* 2016.

[14] Воронцов К.В. Coursera. *Введение в машинное обучение, Бэггинг и случайный лес.* 2016.

[15] Тоби Сегаран. *Программируем коллективный разум.* 2008.

[16] Халафян А.А. *Статистический анализ данных. 3-е издание.* 2007.

[17] Google Places API. *https://developers.google.com/places/?hl=ru.*

[18] KNN. http://www.machinelearning.ru/wiki/index.php?title=knn.

[19] Логистическая регрессия. http://www.machinelearning.ru/wiki/index.php?title=Логистическая_регрессия.

[20] Основы анализа данных на python с использованием pandas и sklearn. https://habrahabr.ru/post/202090/.